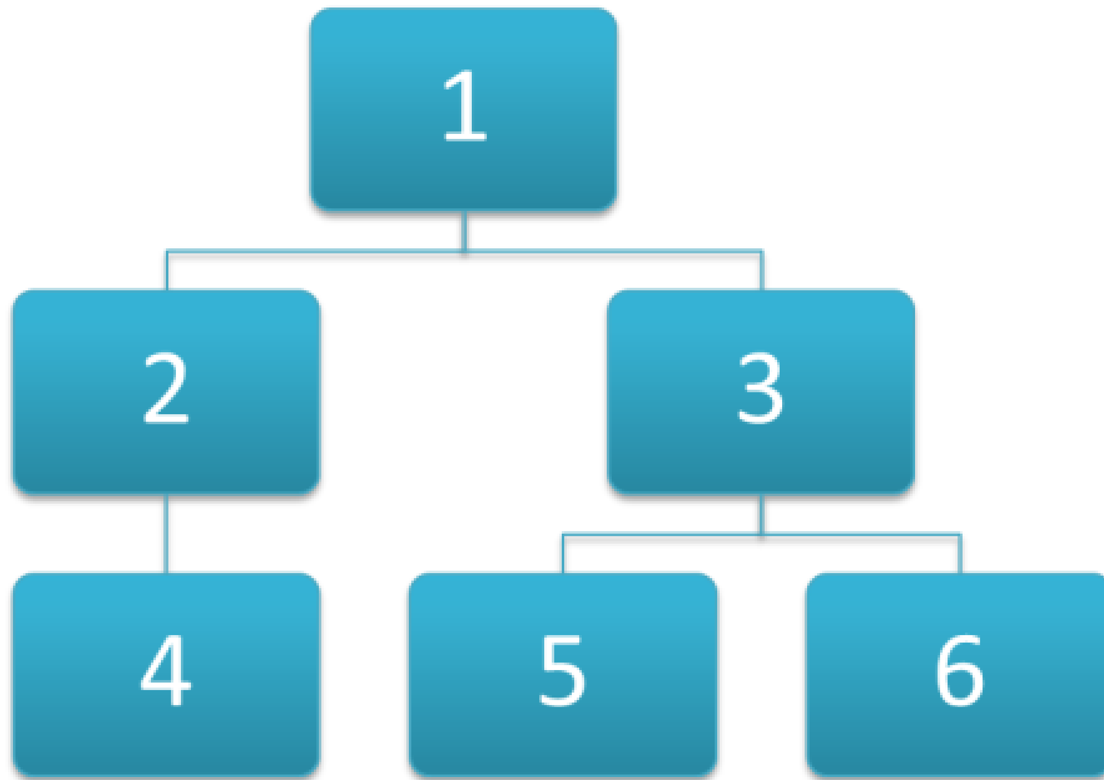
PACMAN

Nombre: Luis González

ÁRBOL



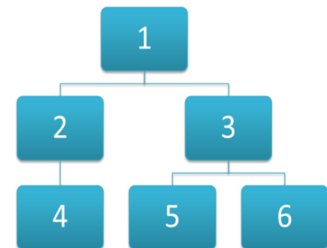
DepthFirstSearch *(last-in-first-out)*

[1]

[1 2] **[1 3]** — **Se extrae el último** (el último; el 3, primero en salir)

[1 2] [1 3 5] **[1 3 6]** — **Adyacentes se agregan al final** *(APPEND)*

```
while not problem.isGoalState(sucesor):
    # Al final estara la trayectoria de un solo nodo
    s, m=cola.pop() # Expande y elimina el ultimo trayecto de la cola
    for sucesor, mov, peso in problem.getSuccessors(s):
        if sucesor not in visitados:
            visitados.append(sucesor)
            # Almacena la ruta expandida y la individual
            rt_act=m+[mov]
            # ALmacena cada ruta
            cola.push((sucesor,rt_act)) # Expandidas se agregan al final
trayecto=rt_act # La ultima trayectoria analizada es la final
return trayecto
```



BreadthFirstSearch *(first-in-first-out)*

[1]

Adyacentes {2} {3}

[1 3] **[1 2]** **Se extrae el último** (el primero; el 2, primero en salir)

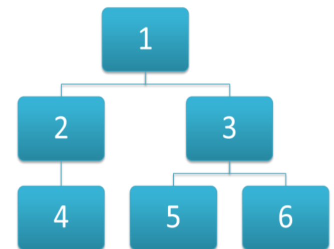
Adyacentes {4}

[1 2 4] **[1 3]** **Adyacentes se agregan al principio** *(INSERT,0)*. **Se extrae el último**

Adyacentes {5} {6}

[1 3 6] [1 3 5] **[1 2 4]**

```
while not problem.isGoalState(sucesor):
    # Al final estaran los que no han sido expandidos
    s, m=cola.pop() # Expande y elimina el ultimo trayecto de la cola
    for sucesor, mov, peso in problem.getSuccessors(s):
        if sucesor not in visitados:
            visitados.append(sucesor)
            # Almacena la ruta expandida y la individual
            rt_act=m+[mov]
            # ALmacena cada ruta
            cola.push((sucesor,rt_act)) # Expandidas se agregan al inicio
trayecto=rt_act # La ultima trayectoria analizada es la final
return trayecto
```



AstarSearch *(prioridad)*

Peso = Costo de la trayectoria + heurística.

Los nodos adyacentes de **menor peso** se expanden, sin importar la ubicación.

[1]

[1 2],peso [1 3],peso

Si [1 2] tiene menor peso, se expande

[1 2 4],peso **[1 3], peso**

[1 3] tiene menor peso

[1 2 4],peso **[1 3 6], peso** [1 3 5], peso

[1 3 6] tiene menor peso

```
while not problem.isGoalState(sucesor):
    s,m=cola.pop() # Expande trayecto de menor peso de la cola
    for sucesor, mov, peso in problem.getSuccessors(s):
        if sucesor not in visitados:
            visitados.append(sucesor)
            # Almacena la ruta expandida y la individual
            rt_act=m+[mov]
            # costo total de la secuencia y el peso heuristico
            cost_rt_act=problem.getCostOfActions(rt_act)+heuristic(sucesor,problem)
            # ALmacena todas las rutas
            cola.push((sucesor,rt_act),cost_rt_act)
trayecto=rt_act # La ultima trayectoria analizada es la final
return trayecto
```